

A fast algorithm for predicting links to nodes of interest

Bolun Chen^{a,b,d}, Ling Chen^{a,c,*}, Bin Li^{a,c}

^a Institute of Information Science and Technology, Yangzhou University, Yangzhou, China

^b College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

^c National Key Lab of Novel Software Tech, Nanjing University, Nanjing, China

^d Department of Physics, University of Fribourg, Chemin du Musée 3, Fribourg CH-1700, Switzerland

The problem of link prediction has recently attracted considerable attention in various domains, such as sociology, anthropology, information science, and computer science. In many real world applications, we must predict similarity scores only between pairs of vertices in which users are interested, rather than predicting the scores of all pairs of vertices in the network. In this paper, we propose a fast similarity-based method to predict links related to nodes of interest. In the method, we first construct a sub-graph centered at the node of interest. By choosing the proper size for such a sub-graph, we can restrict the error of the estimated similarities within a given threshold. Because the similarity score is computed within a small sub-graph, the algorithm can greatly reduce computation time. The method is also extended to predict potential links in the whole network to achieve high process speed and accuracy. Experimental results on real networks demonstrate that our algorithm can obtain high accuracy results in less time than other methods can.

1. Introduction

Many social, biological, and information systems in the real world, from the nervous system to the ecosystem, from road traffic to the Internet, from an ant colony structure to human social relationships, can be naturally described as networks in which vertices represent entities and links denote relationships or interactions between vertices. As a topology approximation of complex systems, due to limitations of time and space, or experimental conditions, it is inevitable that there will be some errors or redundant links in constructing the complex network. At the same time, there will be some undetected potential links. In addition, because of the dynamic evolution of complex network links over time, we must predict missing and potential links according to known network information, which is the goal of the network link-prediction problem [27,32].

The link-prediction problem has a wide range of practical applications in various fields. For example, in biological networks, such as protein-protein interaction networks, metabolic networks and diseases-gene networks [22,43], links existing between nodes indicate that they have an interaction relationship. To mitigate the high costs of biological experimentation to reveal the hidden interaction relationships in these networks, the results of link prediction can direct biological experiments designed to reduce the cost and improve the success rate of the experiments. Predicting the loss and suspicious links of diseases-gene networks can help to explore the mechanisms of diseases, and predict and evaluate their treatment. Furthermore, it can also find new drug targets and open up new paths for drug development [12].

* Corresponding author at: Institute of Information Science and Technology, Yangzhou University, Yangzhou, China. Tel.: +86 514 87870026, fax: +86 514 87887937.

E-mail address: yzulchen@163.com, lchen@yzu.edu.cn (L. Chen).

In social network analysis, link prediction can also be used as a powerful supplementary tool to analyze accurately the social network structure. Studies on online social network analysis have been developing very rapidly in recent years. In online social networks, potential friendship of the users can be revealed by link prediction and can be recommended to the users [34]. By analyzing social relationships, we can find potential interpersonal links [7,9,16,20,21]. Link prediction can also be used in the academic network to predict the type and cooperators of an academic paper [38]. A link-prediction method can also be directly used for information recommendations [25,42] such as a commodity recommendation to customers [24,39]. Marketers would like to recommend products or services based on existing preferences or contacts. Social networking websites customize suggestions for new friends and groups using link prediction. For monitoring e-mail communication, link prediction is applied to detect anomalous e-mail [19]. Financial corporations must monitor transaction networks to detect fraudulent activity via link prediction. In monitoring networks of criminals, link prediction is used to discover hidden connections between criminals to prevent crime or terrorist activity.

Link prediction not only has a wide range of practical value but also has important theoretical significance. For example, link prediction is helpful to understand the mechanism of the evolution of a complex network [23,28,33]. Because the magnitude of the internal characteristics of a complex network structure is very large, it is difficult to compare the advantages and disadvantages of different mechanisms. Link prediction can provide a simple and unified platform for a fair comparison of network evolution mechanisms to promote theoretical research on the complex-network evolution model.

In many real world network applications, we must detect the most-possible links connecting with a given vertex in the network. We must answer queries such as, "Which are the K most-possible links connecting with vertex v in the network?"; "Which five authors share the most-similar research interests with Professor Johnson?"; "Who are the ten customers with the most-similar shopping habits with customer John Smith?"; "What are the closest ten proteins to a given myoglobin?"; and so on. These are actually link-prediction problems on a given vertex. In fact, in many real world applications, we must predict similarity scores only between pairs of vertexes that users are interested in rather than predicting the scores of all pairs of vertexes in the network.

To answer precisely such a link prediction query for a given vertex v using global indices, it is not possible to independently calculate the indices of links connecting with v . Due to the global nature of global indices calculations, we still must calculate the indices of all the node pairs in the network, although we are only interested in the ones involving v . However, calculating the indices of all the node pairs in the network requires a large amount of computation time.

In this paper, we propose a fast similarity-based method to predict links related to the nodes in which users are interested. The method first constructs a sub-graph centered at the node of interest. For a given error bound ε , we can choose the size of such a sub-graph to make the error of the estimated similarities be less than ε . Because the algorithm computes similarity scores only within a small sub-graph, the computation time is greatly reduced. The method is also extended to predict potential links in the whole network and to achieve high process speed and accuracy. Our experiment results on real networks show that the algorithm can achieve higher speed and more accurate results than can other methods.

The rest of this paper is organized as follows. Section 2 reviews related work on link prediction in complex networks. Section 3 reviews methods based on local random walk. Section 4 defines the r -radius sub-graph of a node v in network G for a given error bound ε . Section 5 presents the fast r -radius sub-graph-based algorithm *Single_Node-LP* to predict the links related to the nodes in which the users are interested, and the sub-graph-based algorithm *Node-LP* to predict the links in the whole network. Section 6 shows and analyzes the experimental results obtained by the algorithms *Single_Node-LP* and *Node-LP* and compares their performance with other similar methods. Section 7 presents the conclusions.

2. Related works

In recent years, many methods of link prediction have been reported. Those methods can be classified into three categories: similarity-based methods, machine-learning methods and probabilistic model-based methods.

The similarity-based method is the most commonly used method for link prediction. In the similarity-based method, each node pair is assigned an index, which is defined as the similarity between the two nodes. All non-observed links are ranked according to their similarities, and the non-observed links connecting nodes that are more similar are supposed to have higher existence likelihoods. Node similarity can be defined by using the essential attributes of nodes: two nodes are considered similar if they have many common features or correlated topological structures [1,15,26]. Many studies found that there are substantial levels of topical similarity among individuals who are close to one another in the social network. For instance, Aiello et al. [2] studied friendship prediction in social networks based on the presence of homology in three systems that combine tagging social media with online social networks. Many works exploit topological features of network structures for link-prediction tasks. S. Gao et al. [11] defined the overall relationships between object pairs as a link pattern, which consists of an interaction pattern and a connection structure in the network. The structural similarity indices can be classified into three categories: local indices, global indices, and quasi-local indices. Local indices use only neighbor information of the nodes. Typical local indices include Common Neighbors, the Salton Index, the Jaccard Index, the Sorensen Index, the Hub Depressed Index, the Hub Promoted Index, the Leicht-Holme-Newman Index, the Preferential Attachment Index, the Adamic-Adar Index and the Resource Allocation Index [32]. Global indices require global topological information. The Katz Index, the Leicht-Holme-Newman Index and the Matrix Forest Index [32] are typical global indices. Quasi-local indices do not require global topological information but make use of more information than do local indices. Such indices include the Local Path Index [31,45], Local Random Walk, and Superposed Random Walk [30]. Another similar group is based on the random walk. These include indices such as Average Commute Time, Cos+, Random Walk

with Restart, and SimRank [32]. Lü et al. [31] proposed two new local indices, Resource Allocation index and Local Path index. Empirical results show that these two indices outperform all other local indices. In particular, the local path index, requiring somewhat more information than the common neighbors' index, provides competitively accurate predictions compared with the global indexes. Wang [41] presented a method for predicting link directions using local directed path in a directed network. Liu and Lü [30] studied the link prediction problem based on the Local Random Walk (LRW) and Superposed Random Walk (SRW); they found that the limited step might obtain a better prediction than the result of global random walk. Because the random walk-based methods such as LRW and SRW require $O(n^3)$ time for a network with n nodes, they are impractical in many real applications.

Machine-learning strategies are also exploited in network link-prediction methods. Pujari et al. [36] presented a supervised rank aggregation method for link prediction in complex networks. Vu et al. [40] introduced a continuous-time regression model for network link prediction. The model can incorporate both time-dependent network statistics and time-varying regression coefficients. Zeng et al. [44] presented a method incorporating semi-supervised learning into the link prediction task to use the potential information in a large number of unlinked node pairs in networks. He et al. [14] proposed a link-prediction ensemble algorithm based on an ordered weighted averaging operator. The algorithm assigns weights for nine local information-based link prediction algorithms and then aggregates their results to obtain final prediction scores. Bao et al. [3] advanced a network link predictor using principal component analysis to identify features that are important to link prediction. Bringmann et al. [6] proposed an approach to link prediction in temporal networks based on the techniques of association rules mining and frequent-pattern detecting. Using techniques of data mining and machine learning, the method can predict future co-participation of the individuals in social events. To avoid a high computational cost of optimization in the machine-learning methods, some heuristic methods are employed in link prediction. Sherkat et al. [37] introduces an unsupervised structural link prediction algorithm based on the ant colony optimization. Bliss et al. [5] proposed an approach to predicting future links by applying the covariance matrix-adaptation evolution strategy. Ding et al. [8] advanced a method based on multi-resolution community partitioning to predict potential links in a network.

Some methods for link prediction for a network are based on probabilistic models. Popescul et al. [35] presented a statistical relational learning-based method for link prediction. Hanneke et al. [13] proposed a family of statistical models for social network link prediction by extending the exponential random graph model. Liu et al. [29] presented a method for link prediction in a user-object network. The method considers both time attenuation and diversion delay. S. Gao et al. [10] proposed a model that exploits multiple information sources in the network to obtain link-occurrence probabilities. Barbieri et al. [4] proposed a stochastic topic model for link prediction over directed and nodes-attributed graphs. The model not only predicts links but also produces a different type of explanation for each link predicted. Hu et al. [18] presented a probabilistic model to detect human motion in a social network, and advanced a method for labeling human motion using a constraint-based genetic algorithm to optimize the model. However, such a probabilistic model requires a predefined distribution of link appearances, which is difficult to know in advance for a given network.

3. Local random walk

We consider the network represented by an undirected simple graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Multiple links and self-connections are not considered in G . Let $n = |V|$ be the number of nodes in G . We use U to denote the universal set containing all $n(n-1)/2$ possible links. The task of link prediction is to detect missing links (or links that will appear in the future) in the set of non-existing links $U-E$.

In the similarity-based method, the purpose of link prediction is to assign a score, $S(x, y)$, to each pair of nodes $(x, y) \in U$. This score reflects the similarity between the two nodes. For a node pair (x, y) in $U \setminus E$, a larger $S(x, y)$ is associated with a higher probability that the link between nodes x and y exists.

Liu and Lü [30] studied the link-prediction problem based on the local random walk and found that the limited step may obtain a better prediction than the result of global random walk.

(1) Local Random Walk (LRW)

To measure the similarity between nodes x and y , a particle is initially placed on node x and then walks randomly on the network. A sequence of $n \times n$ matrixes $\pi(t)$ ($t = 0, 1, 2, \dots$) is defined; its element $\pi_{xy}(t)$ is the probability of a particle from x reaching y at time step t . The initial value of the matrix element is defined as

$$\pi_{xy}(0) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

At time t , the particle randomly walks on the network according to transformation matrix $P = [p_{xy}]$, and generates a new matrix $\pi(t+1)$. In transformation matrix P , element p_{xy} indicates the probability that the particle at node x will walk to y in the next step and is defined as $p_{xy} = a_{xy}/d_x$, where a_{xy} is the (x, y) element of the adjacent matrix, and d_x denotes the degree of node x . By the random walk of the particle, the matrix $\pi(t)$ evolves as follows:

$$\pi(t+1) = P^T \pi(t), \quad t \geq 0 \quad (2)$$

The LRW index $s_{xy}^{LRW}(t)$, which measures the similarity between nodes x and y at time step t , is thus defined as the following:

$$s_{xy}^{LRW}(t) = q_x \cdot \pi_{xy}(t) + q_y \cdot \pi_{yx}(t) \quad (3)$$

where q_x and q_y are the initial configuration functions. Liu and Lü [30] suggested a simple form to determine the value of q_x by using the degree node x , namely

$$q_x = \frac{d_x}{2|E|} \quad (4)$$

where $|E|$ is the number of existing links in the network.

(2) Superposed Random Walk (SRW)

Based on the LRW, Liu and Lü [30] proposed the SRW index, in which the particles are continuously released at the starting point, resulting in a higher similarity between the target node and the nodes nearby. Let $S_{xy}^{SRW}(t)$ be the similarity measure between nodes x and y at time step t ; its mathematical expression is as follows:

$$S_{xy}^{SRW}(t) = \sum_{l=1}^t S_{xy}^{LRW}(l) = q_x \sum_{l=1}^t \pi_{xy}(l) + q_y \sum_{l=1}^t \pi_{yx}(l) \quad (5)$$

In each iteration, the time required for calculating matrix $\pi(t)$ is $O(n^3)$ using formula (2), and the time required to calculate the S^{SRW} scores for all pairs of nodes in the network is $O(t.n^3)$. To predict the links connecting a given node v using SRW indices, we still need $O(t.n^3)$ time to calculate the S^{SRW} scores due to the global nature of the SRW indices calculation. Therefore, it is necessary to find a special approach for predicting the links connecting a given node v with less time cost. In this work, we propose a fast similarity-based method to calculate the SRW indices of the links related to the nodes in which the users are interested. Instead of calculating the SRW indices in the whole network, we estimate the SRW indices within a sub-graph centered at the node of interest.

4. Sub-graph of the relevant vertices

In our method, we estimate the SRW indices in a sub-graph centered at the node of interest instead of calculating the SRW indices in the whole network. First, we define the r -radius sub-graph of a given node v in network G .

Definition 1. Let $G = (V, E)$ be the graph representing a network. Denote the shortest path between nodes y and z in G as $shortPath(y, z)$ and its length as $|shortPath(y, z)|$. Let r be a positive integer and $x \in V$ be a node in G . Define the node set $V' \subset V$ as $V' = \{y | y \in V, |shortPath(x, y)| \leq r\}$, which is the set of nodes connecting with x by the shortest path no longer than r . Define the edge set $E' \subset E$ as $E' = \{(y, z) | (y, z) \in E, y, z \in V'\}$. Then, subgraph $G_x(r) = (V', E')$ is called an r -radius sub-graph of node x in network G .

Suppose we need only to predict the links connecting with node x in which the user is interested. To speed the calculation of the similarity scores between x and the other vertexes, our method estimates the SRW indices in the r -radius sub-graph $G_x(r)$. Because only local topological information is used, errors might occur in the SRW indices calculation. However, we can restrict such errors within a narrow limit by setting a proper radius r of sub-graph $G_x(r)$.

Let ε be a given error bound and $x \in V$ be a node in G ; we must construct an r -radius sub-graph $G_x(r)$ such that for every node $y \in G_x(r)$, the SRW indices between y and x must be greater than ε , whereas for all the nodes outside sub-graph $G_x(r)$, their SRW indices with node x must be less than ε . Then, we can neglect the nodes outside sub-graph $G_x(r)$. To construct such a sub-graph, we first present the following theorem to show how to determine a proper radius r such that the nodes outside sub-graph $G_x(r)$ can be neglected in computing the SRW indices involving node x under a given threshold ε .

Theorem 1. Given a threshold $\varepsilon > 0$, a positive integer r and a node $x \in V$, if

$$r \geq \left\lceil \log_2 \frac{q_x + q_{\max}}{\varepsilon} + 1 \right\rceil, \quad (6)$$

then for all $y \in G - G_x(r)$, $S_{xy}^{SRW}(t) \leq \varepsilon$, ($t = 1, 2, \dots$). Here, q_x is the initial configuration function defined in (4), and $q_{\max} = \max_{z \in V} q_z$.

Proof. Let $\pi_{xy}(t)$ be the probability of a particle from node x reaching node y at time step t .

Let $y \in G - G_x(r)$ be a node outside sub-graph $G_x(r)$. By the definition of $G_x(r)$, we know that the length of the shortest path from y to x is greater than r . Thus, the earliest time for the particle starting from node x to reach node y is r . Therefore, $\pi_{xy}(t) = 0$ for $t = 1, 2, \dots, r - 1$.

Now, we investigate the value of $\pi_{xy}(t)$ for $t \geq r$. Obviously, the path from x to y with the maximum possible probability is path l as shown in Fig. 1:

Fig. 1 shows that the degrees of the nodes in path l are all equal to 2 except the two ends. Therefore, the probability of this path is the following:

$$p(l) = \frac{1}{h} \left(\frac{1}{2} \right)^{t-1}$$

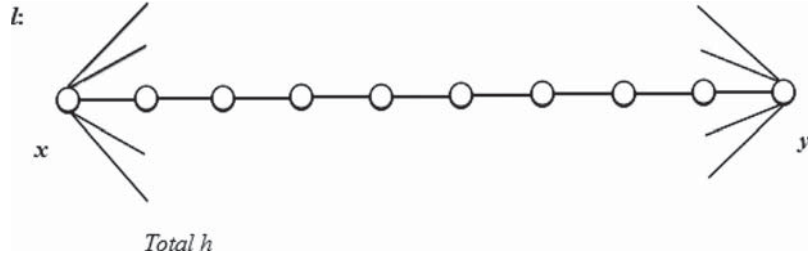


Fig. 1. The path with maximum probability from node x to y .

Here, h is the degree of node x . Let $L(x, y)$ be the set of all paths from node x to y ; we then have

$$\pi_{xy}(t) = \sum_{l \in L(x, y)} p(l) \leq h \cdot \frac{1}{h} \left(\frac{1}{2}\right)^{t-1} = \frac{1}{2^{t-1}}$$

Therefore,

$$\pi_{yx}(t) \leq \frac{1}{2^{t-1}}$$

Because $\pi_{xy}(k) = 0$, for $k = 1, 2, \dots, r-1$, we have the following:

$$\sum_{k=1}^t \pi_{xy}(k) = \sum_{k=r}^t \pi_{xy}(k) \leq \sum_{k=r}^t \frac{1}{2^{k-1}} \leq \frac{\left(\frac{1}{2}\right)^r}{1 - \frac{1}{2}} = \left(\frac{1}{2}\right)^{r-1},$$

and

$$\sum_{k=1}^t \pi_{yx}(k) \leq \left(\frac{1}{2}\right)^{r-1}.$$

By Eq. (5), $S_{xy}^{SRW}(t)$ can be approximated as:

$$S_{xy}^{SRW}(t) \leq q_x \left(\frac{1}{2}\right)^{r-1} + q_{\max} \left(\frac{1}{2}\right)^{r-1} = (q_x + q_{\max}) \left(\frac{1}{2}\right)^{r-1}$$

To estimate the radius of sub-graph $G_x(r)$ such that for every node y outside $G_x(r)$, the SRW indices between y and x must be less than ε , we set $S_{xy}^{SRW}(t) \leq \varepsilon$. Thus, we obtain $(q_x + q_{\max}) \left(\frac{1}{2}\right)^{r-1} \leq \varepsilon$; thus, $\left(\frac{1}{2}\right)^{r-1} \leq \frac{\varepsilon}{q_x + q_{\max}}$.

Therefore, we obtain

$$r \geq \left\lceil \log_2 \frac{q_x + q_{\max}}{\varepsilon} + 1 \right\rceil.$$

□

From Theorem 1, we know that for a given threshold ε and a node x , if r satisfies (6), then indexes $S_{xy}^{SRW}(t)$ of the nodes outside $G_x(r)$ are all less than threshold ε and can be ignored. Therefore, we can predict the links connecting with node x only in sub-graph $G_x(r)$ rather than calculating the SRW values between x and the other nodes in the entire network.

From (6) we can see that a smaller ε value leads to a greater radius r of sub-graph $G_x(r)$ and a higher-quality result. However, a smaller ε value generates a larger sub-graph for a given node, which requires a larger amount of computation time to process it. In the extreme, if we set $\varepsilon = 0$, there is no error allowed in the results. In this case, the radius of the sub-graph is infinite, and $G_x(r)$ is identical to the whole network. Therefore, there is a trade-off between the quality of the results and computational time. We should strike a good balance between them by setting a proper value of ε .

5. Framework of link-prediction algorithms

In this section, we present two link-prediction algorithms based on Theorem 1. One algorithm is for a link-prediction query on the potential links connecting with a given node. The other algorithm predicts links in the whole network.

5.1. Vertex similarity querying algorithm

Given a network $G = (V, E)$, a node x in V , and the error bound ε , our algorithm first computes the radius r of sub-graph $G_x(r)$. After sub-graph $G_x(r)$ is constructed, the value of S_{xy}^{SRW} is calculated for x with every node y in $G_r(x)$.

The framework of our vertex similarity-querying algorithm *Single_Node_LP* (single node link prediction) is as follows.

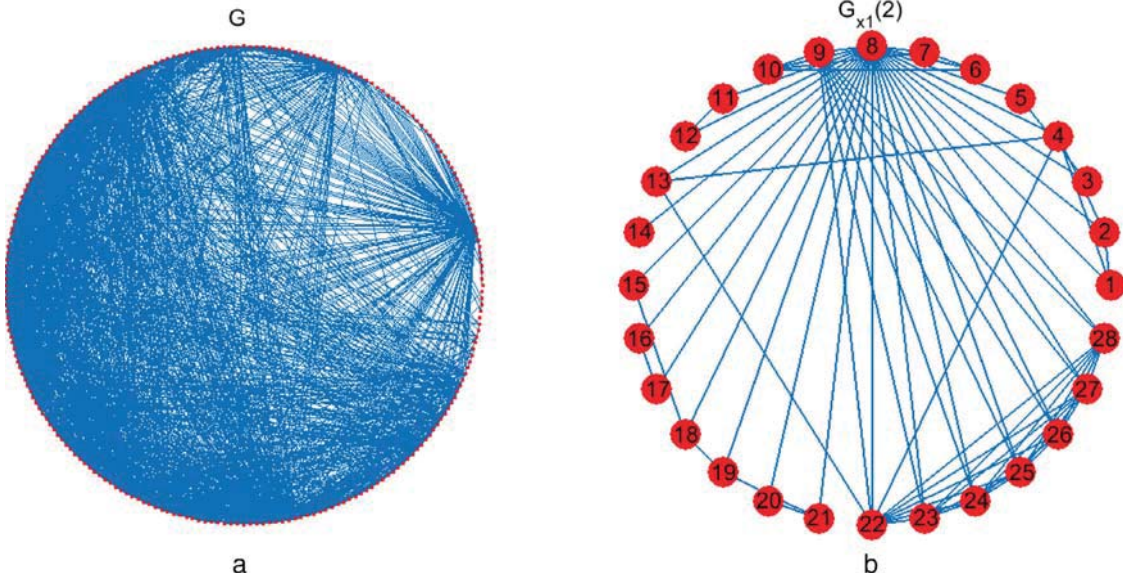


Fig. 2. USAir network and sub-graph $G_{x_1}(2)$.

Algorithm 1 *Single_Node_LP* (x, ε);

Input: A : Adjacency matrix of network $G = (V, E)$;
 ε : The error bound;
 x : The vertex being queried;
Output: The SRW indices between vertex x and the nodes in $G_x(r) = (V_x, E_x)$;
Begin
1. For node x and error bound ε , calculate the radius r of sub-graph $G_x(r)$ according to (6).
2. *generate*($x, r, 0$); /* Generating sub-graph $G_x(r) = (V_x, E_x)$ */
3. **For** every node y in V_x **do**
 Calculate the value of S_{xy}^{SWR} in $G_x(r)$;
4. **Endfor**.
5. Output the score S^{SWR} of the sub-graph;
End

Line 2 of the algorithm generates node set V_x of the sub-graph $G_x(r)$ by calling subroutine *generate*($x, r, 0$). Using the depth-first-search strategy, subroutine *generate*($x, r, 0$) starts from vertex x and recursively searches all the vertices with a distance to x less than r . Initially, all vertices are marked “unvisited”, and the set $V_x = \varphi$. The recursive algorithm *generate*(x, r, t) is as follows.

Algorithm 2 *generate*(x, r, t)

Input: x : The vertex being queried;
 t : The maximum distance between vertex x and other vertices;
Output: V_x : The vertices set in sub-graph $G_x(r)$;
Begin
 $V_x = \Phi$;
If ($t < r$) and (there exist unvisited neighbors of x) **then**
 For each node w that is an unvisited neighbor of x **do**
 $V_x = V_x \cup \{w\}$; Mark w as “visited”;
 generate($w, t + 1$);
 End for
 End if
End

Let d_x be the degree of node x ; then there are at most $r * d_x$ vertices in $G_x(r)$. Therefore, the time complexity of algorithm *generate*(x, r, t) is $O(r * d_x)$. In algorithm *Single_Node_LP*, it takes $O(r^3 * d_x^3)$ time to calculate $\pi(t)$ in $G_x(r)$ by formula (2). In line 3 of *Single_Node_LP*, computing the $S_{xy}^{SWR}(r)$ values in $G_x(r)$ requires $O(r^4 * d_x^3)$ computation time. Therefore, the time complexity of algorithm *Single_Node_LP* is $O(r^4 * d_x^3)$.

To illustrate the procedure of link prediction in our algorithm, we use the US airport network (USAir) dataset as an example.

Example 1. The USAir network shown in Fig. 2(a) has 232 nodes representing the airports in the United States. There are 1635 edges in the network indicating the air routes between the airports. Let x_1 be the vertex being queried, and set $\varepsilon = 0.03$.

First, the algorithm calculates radius r of sub-graph $G_{x_1}(r)$ according to (6) and obtains the result $r = 2$.

Then, subgraph $G_{x_1}(2)$ is constructed as shown in Fig. 2(b). From the figure, we can see that subgraph $G_{x_1}(2)$ has only 28 nodes and 71 edges. We predict the links connecting with node x_1 only in the small sub-graph $G_{x_1}(2)$, instead of calculating the SRW values between x_1 and the other nodes in the entire network of 232 nodes and 1635 edges.

Next, the algorithm calculates the SRW values between x_1 and the other 27 nodes x_2, x_3, \dots, x_{28} in sub-graph $G_{x_1}(2)$.

Finally, we obtain the SRW values $S_{1,j}$ between nodes x_1 and x_j ($j = 2, \dots, 28$). The largest SRW values are $S_{1,8} = 0.8$, $S_{1,4} = 0.76$, $S_{1,2} = 0.75$, and $S_{1,9} = 0.75$.

5.2. Predicting the potential links in the whole network

We also extend the algorithm *Single_Node_LP* and present an algorithm named *Node_LP*(G, ε) to predict the potential links in the whole G network. Algorithm *Node_LP* first computes radius r_x of sub-graph $G_x(r_x)$ for each node x in network G . Then, it constructs sub-graph $G_x(r_x)$ and computes the SWR index for x with every node y in $G_x(r_x)$. Such an SWR score computed in sub-graph $G_x(r_x)$ is denoted $\bar{S}_{xy}^{SWR}(r_x)$. If a node $y \notin G_x(r_x)$ and $x \notin G_y(r_y)$, the value of $\bar{S}_{xy}^{SWR}(r_x)$ is set as ε . The framework of algorithm *Node_LP*(G, ε) is as follows.

Algorithm 3 *Node_LP*(G, ε);

Input: A : Adjacency matrix of network $G = (V, E)$;
 ε : The error bound;
Output: \bar{S}^{SWR} : matrix of similarity indices between all pairs of nodes in G ;
 /* Initial values of all the elements in \bar{S}^{SWR} are set as ε ; */
Begin
 1. **For** every node x in G **do**
 2. Calculate the radius r of sub-graph $G_x(r)$ according to (6) and error bound ε ;
 3. $generate(x, r, 0)$; /*Generating sub-graph $G_x(r) = (V_x, E_x)$ */
 4. **For** every node y in V_x **do**
 If $\bar{S}_{xy}^{SWR} = \varepsilon$ **then**
 $\bar{S}_{xy}^{SWR} = \text{SWR index of node pair } (x, y) \text{ in } G_x(r)$;
 Endif
 5. **Endfor.**
 6. Output the similarity score matrix \bar{S}^{SWR} ;
End

Let d_x be the degree of node x ; then, there are at most $r * d_x$ vertices in $G_x(r)$. Therefore, the time complexity of algorithm *generate*($x, r, 0$) is $O(r * d_x)$. In line 4 of algorithm *Node_LP*, it takes $O(r^3 * d_x^3)$ time to calculate $\pi(t)$ in $G_x(r)$ by formula (2). Therefore, computing the $\bar{S}_{xy}^{SWR}(r)$ values in $G_x(r)$ requires $O(r^4 d_x^3)$ computation time. Let the number of nodes in the whole network be n and time complexity for the link prediction for the whole network by algorithm *Node_LP* be $O(nr^4 d_x^3)$. Here, the value of r depends only on error bound ε and can be treated as a constant for a given value of ε . Because $d_x \leq \max_{v \in V} d_v$ also can be considered a constant for a given network, the time complexity of link prediction on the whole network by algorithm *Node_LP* is $O(n)$. Compared with the method for computing the SRW indices by (5), which requires $O(n^3)$ time, *Node_LP* consumes much less time to obtain high-quality results.

For each node pair (x, y) , if the shortest path between x and y is less than $\min(r_x, r_y)$, \bar{S}_{xy}^{SWR} can be calculated either in $G_x(r_x)$ or in $G_y(r_y)$. To avoid duplicate calculations of \bar{S}_{xy}^{SWR} , the algorithm can construct and process sub-graph $G_x(r_x)$ in the order of the clustering coefficient of node x . The clustering coefficient C_x of node x is defined as:

$$C_x = \frac{2k_x}{d_x(d_x - 1)} \quad (7)$$

Here, k_x is the number of triangles connecting with x , and d_x is the degree of x . If node x has a greater clustering coefficient, its sub-graph $G_x(r_x)$ contains richer topological information. Therefore, to the degree that node x has a larger clustering coefficient, $G_x(r_x)$ should be constructed and processed earlier. \bar{S}_{xy}^{SWR} can be calculated in sub-graph $G_x(r_x)$ if the clustering coefficient of node x is greater than that of node y .

5.3. Error of the estimated SRW value

In algorithm *Node_LP*, we use the similarity score $\bar{S}_{xy}^{SWR}(r)$ in $G_x(r)$ to approximate SRW similarity $S_{xy}^{SRW}(t)$ for node $y \in G_x(r)$. For node y outside $G_x(r)$, we set the value of $\bar{S}_{xy}^{SWR}(r)$ as ε . The following theorem estimates the error of using $\bar{S}_{xy}^{SWR}(r)$ to approximate $S_{xy}^{SRW}(t)$.

Theorem 2. Given an error threshold ε and a node x in $G(V, E)$, if radius r of sub-graph $G_x(r)$ satisfies $r \geq \lceil \log_2 \frac{q_x + q_{\max}}{\varepsilon} + 1 \rceil$, then

$$\left| S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SWR}(r) \right| \leq \varepsilon.$$

Proof.

(1) **Case 1:** $y \in G_x(r)$.

We prove that if $y \in G_x(r)$, then (a) $S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) = 0$ if $t \leq r$; and (b) $S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) \leq \varepsilon$ if $t > r$.

(a) If $t \leq r$, $S_{xy}^{SRW}(t)$ can be calculated in sub-graph $G_x(r)$ directly; it is obvious that $S_{xy}^{SRW}(t) = \bar{S}_{xy}^{SRW}(t)$, and $S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) = 0$.

(b) If $t > r$, because $S_{xy}^{SRW}(t) = q_x \sum_{l=1}^t \pi_{xy}(l) + q_y \sum_{l=1}^t \pi_{yx}(l)$ and $\bar{S}_{xy}^{SRW}(r) = q_x \sum_{l=1}^r \pi_{xy}(l) + q_y \sum_{l=1}^r \pi_{yx}(l)$, we have the following:

$$S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) = q_x \sum_{l=r+1}^t \pi_{xy}(l) + q_y \sum_{l=r+1}^{t-1} \pi_{yx}(l).$$

From the proof of [Theorem 1](#), we know that $\sum_{l=1}^t \pi_{xy}(l) \leq (\frac{1}{2})^{r-1}$. Therefore

$$\sum_{l=r}^t \pi_{xy}(l) \leq \sum_{i=1}^t \pi_{xy}(l) \leq \left(\frac{1}{2}\right)^{r-1}$$

and

$$S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) \leq q_x \left(\frac{1}{2}\right)^{r-1} + q_{\max} \left(\frac{1}{2}\right)^{r-1} = (q_x + q_{\max}) \left(\frac{1}{2}\right)^{r-1}.$$

Because

$$r \geq \left\lceil \log_2 \frac{q_x + q_{\max}}{\varepsilon} + 1 \right\rceil,$$

we have

$$S_{xy}^{SRW}(t) - \bar{S}_{xy}^{SRW}(r) \leq (q_x + q_{\max}) \frac{\varepsilon}{q_x + q_{\max}} = \varepsilon.$$

(1) **Case 2:** $y \notin G_x(r)$.

In this case, the value of $\bar{S}_{xy}^{SWR}(r)$ is set as ε . Because $y \notin G_x(r)$, by [Theorem 1](#), we know that $S_{xy}^{SRW}(t) \leq \varepsilon$. Let $S_{xy}^{SRW}(t) = \varepsilon - \delta$, here $0 \leq \delta \leq \varepsilon$. Therefore, we obtain

$$\left| S_{xy}^{SWR}(t) - \bar{S}_{xy}^{SWR}(r) \right| = |\varepsilon - \delta - \varepsilon| = \delta \leq \varepsilon.$$

□

By [Theorem 2](#), we can see that algorithm *Node_LP* can restrict the error of similarity score $\bar{S}_{xy}^{SRW}(r)$ to be less than threshold ε . Although similarity score $\bar{S}_{xy}^{SRW}(r)$ estimated by *Node_LP* might have an error within the bound ε , it cannot affect the quality of the results because link prediction only depends on the relative ranking of the similarity scores of node pairs rather than their absolute score values.

6. Experimental results

In this section, we empirically demonstrate the effectiveness of proposed algorithms *Node_LP* and *Single_Node_LP* on real world networks. We also compare their performance against other similarity-based link-prediction algorithms. We focus on the accuracy of the results and the algorithms' computing time. All experiments were conducted on the Microsoft Windows 7 operating system, and the results were visualized on Matlab 6.0.

6.1. Datasets

In the experiments, we consider six benchmark datasets [17] representing networks drawn from disparate fields: a protein-protein interaction network (PPI), a co-authorship network between scientists (NS), the electrical power grid of the western US (Grid), a network of US political blogs (PB), the Internet (INT), and a US airport network (USAir). For each dataset, we test its largest connected component. [Table 1](#) summarizes the topological features of the largest components of those networks. In the table, N and M are the total number of nodes and links, respectively. N_C is the number of connected components in the network and the size of the largest one. For example, 1222/2 means that this network has 2 connected components and that the largest one contains 1222 nodes. In the table, e is the efficiency of the network, and C and a are clustering coefficient and assortative coefficient, respectively.

Table 1
Topological features of the giant components in the six networks tested.

Networks	N	M	Nc	e	C	a
USAir	232	1635	232/1	0.440	0.749	-0.228
PB	1224	19090	1222/2	0.397	0.360	-0.221
NS	1461	2742	379/268	0.016	0.798	-0.082
PPI	2617	11855	2375/92	0.180	0.388	0.454
Grid	4941	6594	4941/1	0.056	0.107	0.003
INT	5022	6258	5022/1	0.167	0.033	-0.138

6.2. AUC score

We use AUC (Area Under Curve) scores to evaluate the quality of the results from the algorithms tested. AUC is an important performance measure that has been widely used in many tasks such as cost-sensitive learning, class-imbalance learning, learning to rank, and information retrieval. It can be used in datasets for which traditional criteria such as accuracy and recall are inadequate because AUC is blind to class distribution. AUC also can be used as a measurement to evaluate the accuracy of link-prediction results. The similarity-based link-prediction methods assign a score to each existing and non-existing edge in the network. The AUC value is only the area under the ROC (receiver operating characteristic) curve. In general, a larger AUC value indicates higher performance; hence, the AUC value of the perfect result is 1.0, whereas the AUC of a result by a random predictor is 0.5.

Suppose there are n_1 existing links and n_2 non-existing links in the network, and $n = n_1 + n_2$. Let $\{e_1, e_2, \dots, e_{n_1}\}$ and $\{e_{n_1+1}, e_{n_1+2}, \dots, e_n\}$ be the sets of existing and non-existing links, respectively, and s_i be the score assigned to e_i , ($i = 1, 2, \dots, n$). The ROC curve for the link-prediction result can be drawn as follows:

- (1) Sort the links in descending order of their scores. Let the sorted sequence of the edges be e'_1, e'_2, \dots, e'_n .
- (2) Create an ROC coordinate system in which the abscissa represents the non-existing edges and the ordinate represents the existing edges. The abscissa consists of n_2 steps, and the length of each step is $1/n_2$. The ordinate consists of n_1 steps, and the length of each step is $1/n_2$. Therefore, the ROC coordinate system forms a square with unit side length.
- (3) The ROC curve starts at the origin of the coordinate system and then moves upwards and rightwards towards the upper-right corner of the square. The direction of the movement in each step depends on the sorted sequence e'_1, e'_2, \dots, e'_n . In the i th step, if e'_i is an existing edge, then the curve moves upwards one step. If e'_i is a non-existing edge, e'_{i+1} is an existing edge and $s'_i = s'_{i+1}$. Then, the curve moves one step along the diagonal; otherwise it moves rightwards one step. When the curve reaches the upper-right corner of the square after n steps, it forms the ROC curve.
- (4) The area of the part in the square under the ROC is just the AUC value of the link prediction result.

We illustrate the procedure of creating an ROC curve using the following example.

Example 2. Suppose there are 7 links named a, b, c, d, e, f, g in the network, and a, b, c, d are existing links, whereas e, f, g are the non-existing ones. A predicting result assigns scores $s_a, s_b, s_c, s_d, s_e, s_f, s_g$ to the corresponding links; $s_a > s_e = s_b > s_f > s_c > s_d > s_g$. To create the ROC curve for this result, we first sort the links in descending order of their scores and obtain the sorted sequence a, e, b, f, c, d, g . Then, an ROC coordinate system is formed as shown in Fig. 3, in which the abscissa consists of 3 steps, and the ordinate consists of 4 steps. Starting at the origin of the coordinate system, the curve initially moves upwards one step because the first link in the sequence is a , which already exists. Then, the curve moves one step along the diagonal because the next two links in the sequence are non-existing link e and existing link b ; $s_e = s_b$. The next link in the sequence is f , which is a non-existing one; thus, the curve moves one step rightwards. Repeating such movement until reaching the upper-right corner of the square, the ROC curve is completed as shown in Fig. 3. The area of the shaded part under the curve is the AUC of the result. From the figure, we can see that the AUC is $7.5/12 = 0.625$.

In fact, we need not draw the ROC curve to calculate the AUC value of a result. The AUC value can be interpreted as the probability that a randomly chosen existing link is given a higher score than is a randomly chosen non-existing link. We can randomly pick an existing link and a non-existing one to compare their similarity scores. If, among n independent comparisons, there are n' times the existing link having a higher score and n'' times they have the same score, the AUC value is

$$AUC = (n' + 0.5n'')/n \quad (8)$$

We use the following examples to illustrate the calculation of AUC according to (8).

Example 3. We estimate the AUC of the predicted results on the USAir dataset in Example 1. From Fig. 2(b), we can see that in subgraph $G_{x1}(2)$, node x_1 only connects with nodes x_2, x_4 and x_8 . Therefore, there are 3 existing links, namely (x_1, x_2) , (x_1, x_4) and (x_1, x_8) , and 24 nonexistent links. To compute the AUC of the results, we compare the similarity score of an existing link with that of a non-existing one. Because there are $3 \times 24 = 72$ pairs of existing and non-existing links, the number of comparisons is $n = 72$. Because the 3 exiting links (x_1, x_2) , (x_1, x_4) and (x_1, x_8) have the highest SRW values $S_{18} = 0.8$, $S_{14} = 0.76$, $S_{12} = 0.75$, their SRW values are all greater than or equal to that of the non-existing one. Among 72 such comparisons, existing and non-existing

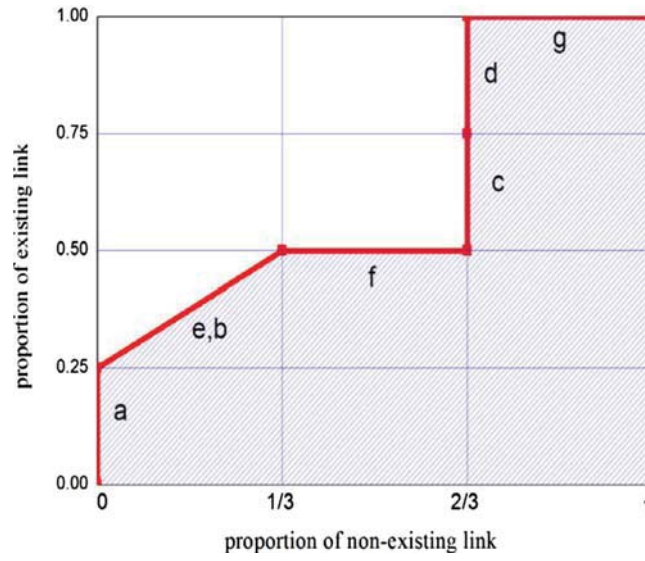


Fig. 3. ROC curve of Example 2.

Table 2

Comparison of AUC scores by *Node_LP* with other methods (best values in bold).

	USAir	PB	NS	PPI	Grid	INT
CN	0.939	0.926	0.987	0.916	0.638	0.650
Salton	0.926	0.878	0.975	0.923	0.612	0.647
Jaccard	0.899	0.865	0.980	0.920	0.622	0.657
Sorensen	0.917	0.885	0.985	0.917	0.633	0.642
HPI	0.840	0.861	0.983	0.910	0.635	0.651
HDI	0.890	0.876	0.980	0.921	0.632	0.652
LHN_I	0.727	0.754	0.972	0.910	0.626	0.650
PA	0.896	0.908	0.671	0.854	0.577	0.959
LP	0.932	0.929	0.985	0.966	0.696	0.942
Katz	0.934	0.942	0.989	0.969	0.956	0.974
Node_LP	0.947	0.995	0.991	0.986	0.961	0.981

links have equal SRW values only when the SRW value of (x_1, x_2) is compared with that of (x_1, x_9) . Therefore, $n' = 71$, $n'' = 1$, and $AUC = (71 + 0.5)/72 = 0.993$.

Example 4. We estimate the AUC of the predicting results in Example 2. In this example, there are 4 existing links and 3 non-existing ones; thus, scores of $3 \times 4 = 12$ pairs of existing and non-existing links must be compared. Among the 12 pairs, the existing link has a greater score than the non-existing one in 7 pairs: (a, e) , (a, f) , (a, g) , (b, f) , (b, g) , (c, g) and (d, g) . Only existing link b has a score equal to that of a non-existing link, e . Therefore, $n' = 7$, $n'' = 1$ and $AUC = (7 + 0.5)/12 = 0.625$.

6.3. Test on the quality of the results by *Node_LP*

In the experiments for testing algorithm *Node_LP*, we divide the set of links in the network into training set and test set and then construct the sub-graph of each vertex based on the training set. We use AUC scores to evaluate the quality of the results from the algorithms tested.

To evaluate the accuracy of the results, a random 10-fold cross validation (CV) is used. In 10-fold cross validation, the original links are randomly partitioned into 10 subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the algorithms, and the remaining 9 subsets are used as training data. The cross-validation process is then repeated 10 times. The 10 results from the folds are averaged to produce a single estimation. We also compare the performance of *Node_LP* against the other similarity-based link-prediction algorithms such as common neighbor (CN), Salton, Jaccard, Sorensen, Hub Promoted Index (HPI), Hub Depressed Index (HDI), Leicht-Holme-Newman Index (LHN_I), Preferential Attachment (PA), Local Path (LP) and Katz [32]. Table 2 presents the average AUC scores on 10-fold CV tests by different algorithms. In the table, the highest AUC scores for the datasets by the 11 algorithms are emphasized in boldface.

As shown in Table 2, among all the 11 algorithms, *Node_LP* has the highest AUC scores on all of the datasets. For example, on the dataset USAir, algorithm *Node_LP* obtains an AUC score of 0.947 after only 4 steps. This shows that algorithm *Node_LP* can achieve

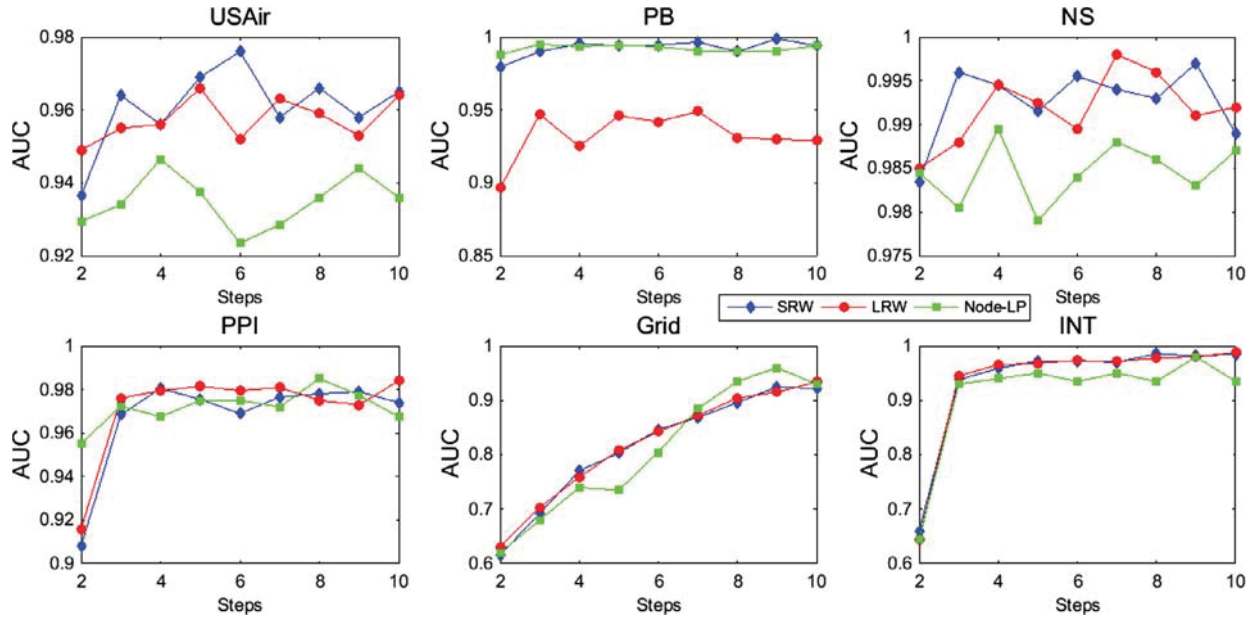


Fig. 4. AUC score in each step of *Node_LP*, LRW and SRW.

Table 3

AUC scores of the results under different ε (best values in bold).

USAir	ε	0.03	0.04	0.05	0.06	0.07
PB	AUC	0.957	0.795	0.627	0.542	0.500
	ε	0.01	0.011	0.012	0.013	0.014
NS	AUC	0.913	0.889	0.773	0.666	0.594
	ε	0.01	0.02	0.03	0.04	0.05
PPI	AUC	0.994	0.921	0.539	0.502	0.500
	ε	0.002	0.004	0.006	0.008	0.010
Grid	AUC	0.976	0.923	0.755	0.599	0.503
	ε	0.00002	0.0001	0.0002	0.0004	0.0008
INT	AUC	0.8850	0.855	0.795	0.715	0.680
	ε	0.002	0.004	0.006	0.008	0.010
	AUC	0.965	0.950	0.695	0.645	0.615
	ε	0.002	0.004	0.006	0.008	0.010

high-quality results. Comparing Tables 1 and 2 shows that the AUC scores by the algorithms are roughly proportional to the clustering coefficients of the dataset tested. Generally, an algorithm can obtain better results on datasets with larger clustering coefficients. However, algorithm *Node_LP* still achieves better results on the networks with lower clustering coefficients. This shows that algorithm *Node_LP* can achieve strong robustness.

We also test the algorithm on six datasets and compare the AUC values in different steps of *Node_LP* with those of the LRW and SRW methods. The results are shown in Fig. 4.

Fig. 4 shows that all three algorithms can obtain a better result when the steps number more than 2. However, in most of the datasets, algorithm *Node_LP* can achieve the best performance after 8 to 10 steps. It should be noticed that *Node_LP* detects the potential links only in a small sub-graph around each node, whereas LRW and SRW perform a global detection in the whole network. The reason for algorithm *Node_LP* achieving high-quality results is that it predicts the potential links within a sub-graph in which the important topological features of the original network are well maintained and the nonessential features are ignored. Although the similarity score by *Node_LP* may have an error within a bound ε , it cannot affect the quality of the results because link prediction only depends on the relative ranking of the similarity scores on node pairs rather than their absolute values.

We also investigate the relationship between the AUC scores of the results and the values of error bound ε . We test algorithm *Node_LP* with different ε values on six datasets. The AUC scores on different datasets are shown in Table 3. In the table, the highest AUC scores for the datasets by different ε values are emphasized in boldface.

From Table 3, we can see that the AUC score decreases when a greater error-bound ε is set. With a lower error-bound, algorithm *Node_LP* generates a smaller sub-graph for each node and can reduce the computation time. We should strike a good balance between the computation time and the quality of the result by setting a proper value of ε . However, such a proper value of ε is problem dependent. It remains an open problem how to set a proper value of ε for a given dataset. In our experiments, we

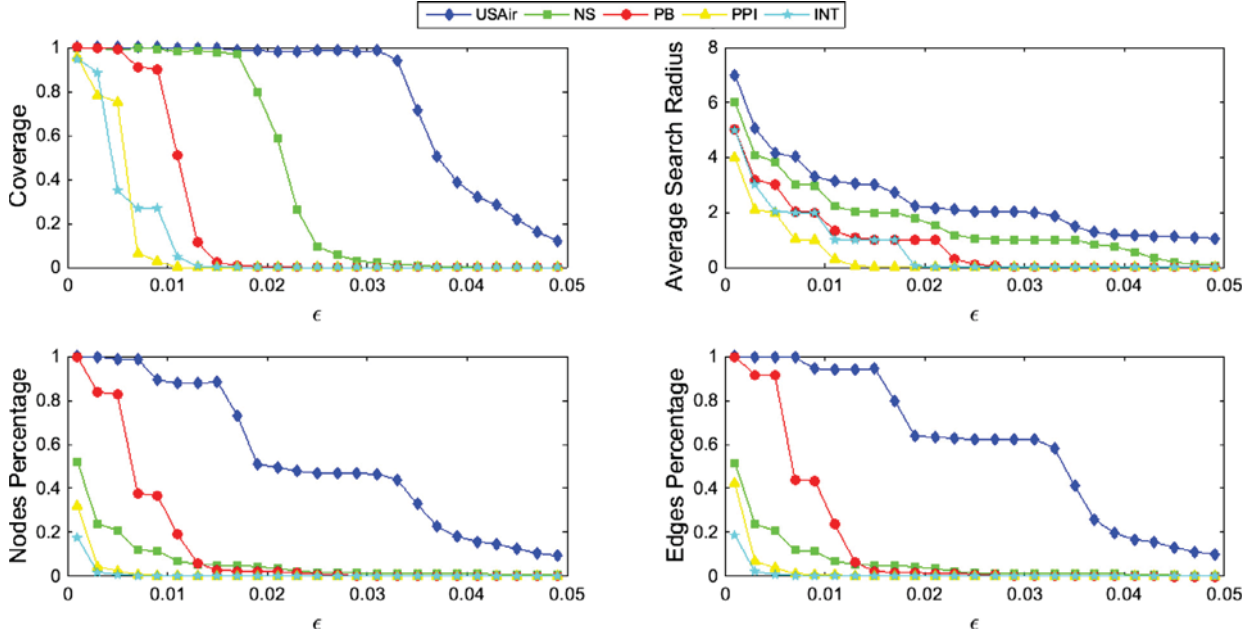


Fig. 5. The coverage, average search radius, node and edge percentage under different ϵ values.

set the value of ϵ for a given dataset by considering the size of the network. For a network of larger size, we set a smaller ϵ value to obtain a greater radius r and cover more-global information. For instance, because network USAir has a smaller size than INT, we set a greater ϵ value for USAir (0.03 to 0.07) and assign a smaller ϵ value for INT (0.002 to 0.01) in the experiments shown in Table 3.

6.4. Test on quality of the results of Single_Node_LP

Next, we test the local link-prediction algorithm $Single_Node_LP(x, \epsilon, K)$, which detects the links connecting with a given node x . In the experiment, we select 90% of the links as a training set denoted E_{train} and the other 10% of the links as a test set denoted E_{test} . We construct sub-graph $G_x(r) = (V_x, E_x)$ for vertex x based on the training set. We define the nodes having direct connection with x as the first-order neighbors of x . Let $\Gamma_r(x)$ be the set of all r -order neighbors of node x in the network, and $\Gamma_{1,r}(x) = \bigcup_{i=1}^r \Gamma_i(x)$. To measure the percentage that sub-graph $G_x(r) = (V_x, E_x)$ covers the nodes in the original network $G = (V, E)$, we define the coverage of $G_x(r) = (V_x, E_x)$ as $coverage(x, r) = \frac{|V_x|}{|\Gamma_{1,r}(x)|}$ and call it r -hop coverage. Obviously, a low value of $coverage(x, r)$ indicates that using $G_x(r)$ can only predict a limited number of objects; thus, it has less significance to the target users. In link prediction, coverage has great importance because only a high coverage could provide a sufficient number of objects for the target user.

In the experiments, we test the coverage of algorithm $Single_Node_LP(x, \epsilon, K)$ under different ϵ values on five datasets. Fig. 5 shows the global coverage of algorithm $Single_Node_LP(x, \epsilon, K)$. The global coverage is defined as the average coverage of all the vertexes tested. Fig. 5 also shows the average search radius and the average number of nodes and links in the sub-graphs under different ϵ values. A smaller search radius and percentage of nodes and links indicate that algorithm $Single_Node_LP(x, \epsilon, K)$ can obtain high-quality link-prediction results in a smaller scale of sub-graph.

Fig. 5 shows that all of the tested values present a downward trend with an increment of the ϵ value. For example, when $\epsilon = 0.034$, the coverage is 0.9312, the average search radius is 1.89, the percentage of nodes in the sub-graph is 47.4% and the percentage of links is 43.8% in the dataset of USAir. Concerning the dataset of INT, when $\epsilon = 0.003$, the coverage is 0.887, the average search radius is 2.00, the percentage of nodes is 1.86% and the percentage of links is 2.23%. From the figure, we can see that algorithm $Single_Node_LP$ can obtain high coverage within small sub-graphs.

We also test the algorithm $Single_Node_LP$ with different coverage values. For an r -hop coverage δ , the sub-graph $G_x = (V_x, E_x)$ of a given node x can be constructed as follows. Let $\Gamma_{1,r}(x)$ be the set of the first- to the r th- order neighbors of x in the training set. We randomly select $\delta \cdot |\Gamma_{1,r}(x)|$ nodes to form set V_x , and the links within V_x in the training set form set E_x .

We test the average numbers of the first- and second-order neighbors in the sub-graphs detected by algorithm $Single_Node_LP$ with different coverage values. The results are shown in Table 4. In the table, we also compare the average numbers of the first- and second-order neighbors with those obtained for the whole network.

As shown in Table 4, we can see that the average numbers of the first- and second-order neighbors of each vertex in a sub-graph are greater than those of the whole network when the coverage is greater than 0.9. This is because the non-observed links

Table 4

Average numbers of the first- and second-order neighbors under different coverages.

Coverage	first-order neighbors				second-order neighbors			
	0.9	0.8	0.7	whole network	0.9	0.8	0.7	whole network
USAir	16.718	16.131	15.558	14.095	108.851	112.455	113.503	116.060
PB	29.145	29.934	30.112	27.355	390.749	409.949	416.943	485.087
NS	4.284	4.281	4.270	4.823	18.296	18.127	17.876	21.016
PPI	12.161	13.303	14.746	9.847	71.580	76.492	68.600	66.149
Grid	2.415	2.415	2.409	2.669	7.187	7.360	7.399	8.046
INT	2.710	2.655	2.476	2.492	36.684	37.087	37.553	27.419

Table 5

Average topological features of the sub-graphs tested.

Networks	ε	N	M	e	C	a
USAir	0.033	96.700	808.317	0.581	0.670	-0.334
PB	0.010	448.588	6537.141	0.407	0.389	-0.222
NS	0.017	18.995	40.686	0.629	0.684	-0.394
PPI	0.002	273.732	1664.373	0.386	0.421	-0.037
Grid	0.000002	421.472	508.813	0.145	0.049	-0.239
INT	0.003	95.886	129.935	0.476	0.053	-0.718

Table 6

AUC scores of the results under different coverage values (best values in bold).

	Coverage = 0.9	Coverage = 0.8	Coverage = 0.7	Coverage = 0.6	Coverage = 0.5
USAir	0.947	0.908	0.882	0.867	0.870
PB	0.995	0.926	0.907	0.869	0.832
NS	0.990	0.949	0.927	0.887	0.828
PPI	0.985	0.940	0.925	0.875	0.855
Grid	0.960	0.840	0.830	0.810	0.790
INT	0.980	0.926	0.870	0.805	0.755

with very low probability of occurrence are ignored by algorithm *Single_Node_LP*, and only the node pairs with rich topological information can be retained. In other words, the potential links recommended by algorithm *Single_Node_LP* have a higher possibility of existence than do those detected in the whole network, overall. Therefore, link-prediction results by algorithm *Single_Node_LP* are reliable and accurate.

In another test, we fix the coverage at 0.9 and choose the optimal ε value of each dataset as indicated in Fig. 5. We then calculate the topological properties of sub-graph $G_x(r)$ generated by algorithm *Single_Node_LP*(x, ε, K). The topological features of the generated sub-graphs are shown in Table 5.

Comparing the topological features of the original networks shown in Table 1, we can see from Table 5 that after narrowing the range of each set of data, the generated sub-graph $G_x(r)$ for each node x has a significant decrease in the average number of vertices and edges; the efficiency e of the network is also significantly improved. The clustering coefficient C of the network does not change significantly, the distribution coefficient of the network becomes smaller, and the average degree of the network increases slightly, indicating that the sub-graphs built in the experiments by algorithm *Single_Node_LP* still maintain the topological characteristics of the original network.

We also investigate the relationship between the AUC value and coverage; we test the AUC of the results with algorithm *Single_Node_LP* by setting the coverage values as 0.9, 0.8, 0.7, 0.6 and 0.5. The experimental results are shown in Table 6. In the table, the highest AUC score on each dataset by different coverage is emphasized in boldface.

From Table 6, we can see that the AUC score decreases when the coverage is reduced. When the coverage exceeds 0.8, *Single_Node_LP* can obtain AUC scores greater than 0.9. However, the value of coverage strongly depends on error bound ε . From Fig. 5, we can observe that the coverage can be greater than 85% when we set the value of ε as 0.01. If we set the value of ε as 0.005, then the coverage can exceed 90% in most of the datasets. Table 6 shows that if the coverage is greater than 90%, algorithm *Single_Node_LP* can obtain high AUC values greater than 0.95. This indicates that algorithm *Single_Node_LP* can obtain high-quality results, with a tolerable error less than ε .

6.5. Test on the time requirement of the algorithms

Computational complexity is another important concern in the designing of a link-prediction algorithm. In the experiments, we compared the time complexity of algorithm *Node_LP* with those of algorithms based on indexes CN, Jaccard, Sorensen, HDI, LHN_I, PA, LP, LRW, SRW and Katz. Fig. 6 shows the comparison of the average computation times required by different algorithms. The figure shows that algorithm *Node_LP* consumes much less computational time than do the other algorithms.

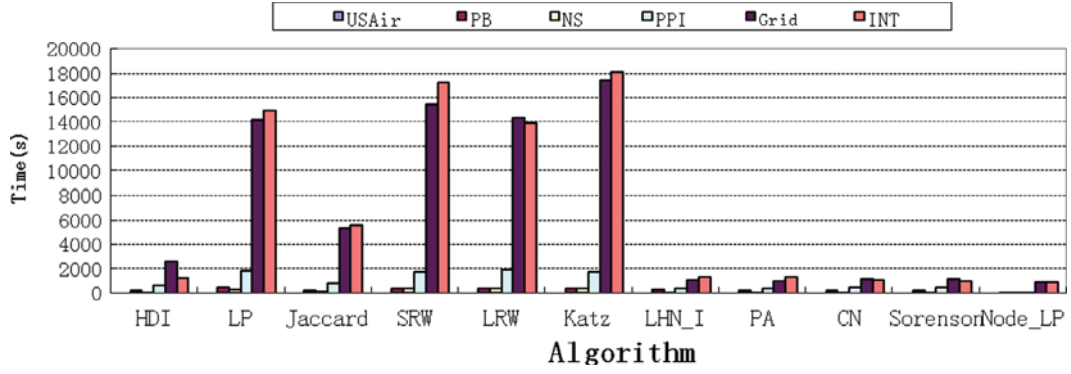


Fig. 6. Running time of the algorithms.

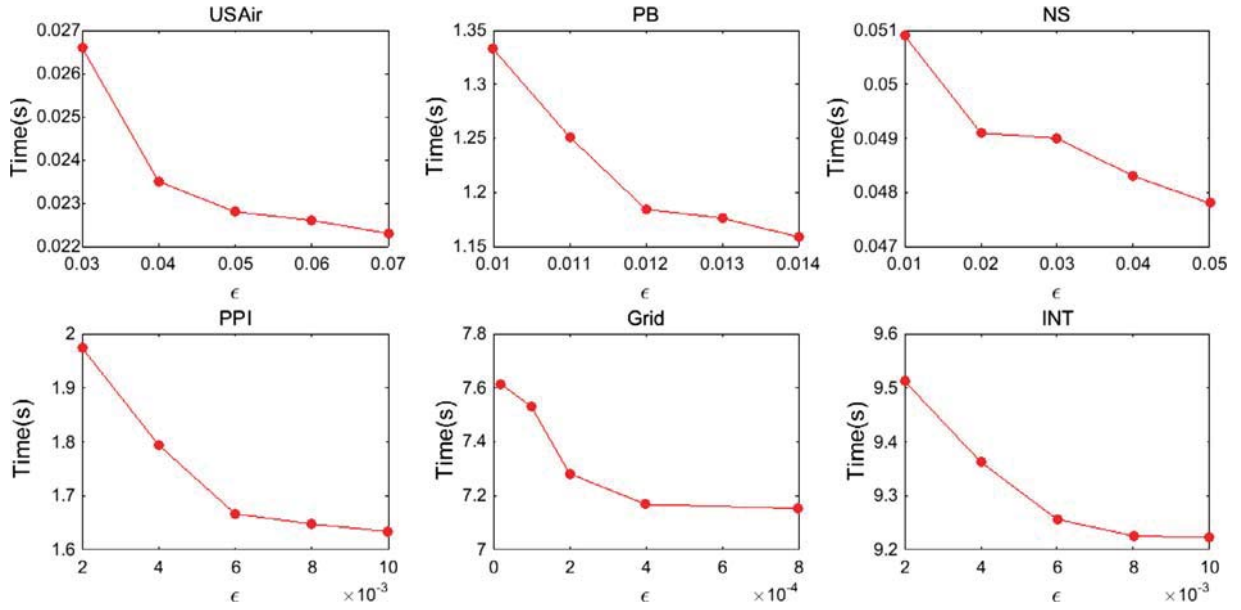


Fig. 7. Computational time required under different ε values.

Let n be the number of nodes in the network and k be the average degree of the nodes; then, time complexity for link prediction in the network by the *Node_LP* algorithm is $O(n)$. For local indexes such as CN, computing the common neighbors for each node pair (v_i, v_j) takes $O(k)$. Therefore, the time complexity for computing the CN index for all of the node pairs is $O(n^2k)$. Other local indexes such as Jaccard, Sorensen, HDI, LHN_I and PA also have a similar time complexity of $O(n^2k)$. Because all of those local similarity indexes use only the first-order neighbors of the nodes, their time complexities are $O(n^2)$. Obviously, computing these indexes requires much more time than *Node_LP*. Because these local indexes use only topological information of first-order neighbors, the quality of their prediction results are much lower than that of *Node_LP*. Similar to *Node_LP*, LP, LRW and SRW are quasi-global random walk-based indexes. The time complexity to compute the LP index is $O(nk^3)$ if we treat the number of steps as a constant. The LRW and SRW indices need $O(n^3)$ time to calculate matrix $\pi(t)$ in LRW and $O(t.n^3)$ time to calculate the S^{SRW} scores. To compute a global topological path-based index such as Katz, the time complexity is $O(n^3)$. Compared with all those similarity-based methods, the *Node_LP* algorithm can achieve high-quality prediction results in less computational time. The reason for *Node_LP* consuming less computational time is that it considers structure information based on the local random walk; thus, it can obtain a better prediction result in fewer time steps than can the global random walk and other local random walk-based indexes.

We also test the computational time required by *Node_LP* under different ε values. Fig. 7 shows the results on different datasets. From the figure, we can see that for each dataset, using a smaller ε value leads to increased computation time cost. The reason is that a smaller ε value implies a higher accuracy requirement. To obtain more-precise results, the algorithm must generate larger-sized sub-graphs, which consumes more computation time.

7. Conclusions and further work

With the large amount of network data available in electric form today, link prediction has become a popular subarea in data mining. We have presented a method for predicting links connected with a given node in which the user is interested. The method first constructs a sub-graph centered at the node of interest. By choosing a proper size of such a sub-graph, we can restrict the error of the estimated similarities within a given threshold. Because the similarity score is computed in a small sub-graph, the algorithm greatly reduces the computation time. We also have extended the method to predict potential links in the whole network to achieve high process speed and accuracy. Although the algorithm is based on a quasi-global random walk approach, it requires $O(n)$ time to obtain high quality results, whereas computing other quasi-global indexes requires $O(n^3)$ time. Experimental results on real networks have shown that our algorithm can obtain higher-accuracy results in less time than other methods can.

One limitation of our algorithm is that a proper ε value must be chosen to strike a good balance between the quality of the results and the computational time. However, such a proper value of ε highly depends on the dataset and the application. Generally, for a network of larger size, we set a smaller ε value to obtain a greater radius r and cover more-global information. For applications requiring high processing speeds, ε should be assigned a larger value. For applications requiring high accuracy of results, a smaller ε value should be set. In our future work, we intend to find an efficient method of obtaining the optimal value of ε for a given dataset.

Acknowledgments

This research was supported in part by the Chinese National Natural Science Foundation under grant Nos. 61379066, 61070047, 61379064, 61472344, and 61402395, the Natural Science Foundation of Jiangsu Province under contracts BK20130452, BK2012672, BK2012128, and BK20140492, the Natural Science Foundation of the Education Department of Jiangsu Province under contracts 12KJB520019, 13KJB520026, and 09KJB20013, the Six-talent peaks project in Jiangsu Province (Grant No. 2011-DZXX-032), and the Foundation of Graduate Student Creative Scientific Research of Jiangsu Province under contract CXZZ13_0172. The China Scholarship Council also supported this work.

References

- [1] M.W. Ahn, W.S. Jung, Accuracy test for link prediction in terms of similarity index: the case of WS and BA models, *Phys. A: Stat. Mech. Appl.* 429 (1) (2015) 177–183.
- [2] L.M. Aiello, A. Barrat, R. Schifanella, et al., Friendship prediction and homophily in social media, *ACM Trans. Web (TWEB)* 6 (2) (2012) 9.
- [3] Z.F. Bao, Y. Zeng, Y.C. Tay, sonLP: social network link prediction by principal component regression, in: *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013.
- [4] N. Barbieri, F. Bonchi, G. Manco, Who to follow and why: link prediction with explanations, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 1266–1275.
- [5] C.A. Bliss, M.R. Frank, Christopher M. Danforth, Peter Sheridan Dodds, An evolutionary algorithm approach to link prediction in dynamic social networks, *J. Comput. Sci.* 5 (5) (2014) 750–764.
- [6] B. Bringmann, M. Berlingerio, F. Bonchi, A. Gionis, Learning and predicting the evolution of social networks, *IEEE Intell. Syst.* (2010) 26–34.
- [7] F. Buccafurri, G. Lax, A. Nocera, D. Ursino, Discovering missing edges across social networks, *Informat. Sci.* 319 (2015) 18–37.
- [8] J.Y. Ding, L.C. Jiao, J.S. Wu, Y.T. Hou, Y.T. Qi, Prediction of missing links based on multi-resolution community division, *Phys. A: Stat. Mech. Appl.* 417 (1) (2015) 76–85.
- [9] J. Fournet, A. Barrat, Contact patterns among high school students, *PLoS ONE* 9 (9) (2014) e107878.
- [10] S. Gao, L. Denoyer, P. Gallinari, Temporal link prediction by integrating content and structure information, in: *Proceedings of CIKM'11, Glasgow, Scotland, UK*, 2011, pp. 1169–1174.
- [11] S. Gao, L. Denoyer, P. Gallinari, et al., Probabilistic latent tensor factorization model for link pattern prediction in multi-relational networks, *J. China Univ. Posts Telecommun.* 19 (2012) 172–181.
- [12] R. Guimera, M. Sales-Pardo, Missing and spurious interactions and the reconstruction of complex networks, *Proc. Nat. Acad. Sci., USA* 106 (52) (2010) 22073–22078.
- [13] S. Hanneke, W.J. Fu, E.P. Xing, Discrete temporal models of social Networks, *Electron. J. Stat.* 4 (2010) 585–605.
- [14] Y.L. He, James, N.K. Liu, Yan-xing Hu, Xi-zhao Wang, OWA operator based link prediction ensemble for social network, *Expert Syst. Appl.* 42 (1) (2015) 21–50.
- [15] M. Hoffman, D. Steinley, M.J. Brusco, A note on using the adjusted Rand index for link prediction in networks, *Social Netw.* 42 (2015) 72–79.
- [16] T. Hossmann, G. Nomikos, T. Spyropoulos, F. Legendre, Collection and analysis of multi-dimensional network data for opportunistic networking research, *Comput. Commun.* 35 (13) (2012) 1613–1625.
- [17] <http://www.linkprediction.org/index.php/link/resource/data> (accessed in July 2014).
- [18] F.Y. Hu, H.S. Wong, Labeling of human motion based on CBGA and probabilistic model, *Int. J. Smart Sens. Intell. Syst.* 6 (2) (2013) 583–609.
- [19] Z. Huang, D.K.J. Lin, The time-series link prediction problem with applications in communication surveillance, *INFORMS J. Comput.* 21 (2009) 286–303.
- [20] N.M.A. Ibrahim, L. Chen, Link prediction in dynamic social networks by integrating different types of information, *Appl. Intell.* 42 (4) (2015) 738–750.
- [21] K. Jahanbakhsh, V. King, Gholamali C. Shoja, Predicting missing contacts in mobile social networks, *Pervasive Mobile Comput.* 8 (5) (2012) 698–716.
- [22] B. Kaya, M. Poyraz, Age-series based link prediction in evolving disease networks, *Comput. Biol. Med.* 63 (2015) 1–10.
- [23] B. Kaya, M. Poyraz, Supervised link prediction in symptom networks with evolving case, *Measurement* 56 (2014) 231–238.
- [24] X. Li, H.C. Chen, Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach, *Decis. Support Syst.* 54 (2) (2013) 880–890.
- [25] J. Li, L.L. Zhang, F. Meng, F.H. Li, Recommendation algorithm based on link prediction and domain knowledge in retail transactions, *Procedia Comput. Sci.* 31 (2014) 875–881.
- [26] H. Liao, A. Zeng, Y.C. Zhang, Predicting missing links via correlation between nodes, *Physica A: Stat. Mech. Appl.* 436 (2015) 216–223.
- [27] R.N. Lichtenwalter, New precepts and method in link prediction, in: *Proceedings of ACM KDD'*, Vol. 10, 2010, pp. 243–252.
- [28] H.K. Liu, L.Y. Lü, T. Zhou, Uncovering the network evolution mechanism by link prediction (in Chinese), *Sci. Sin. Phys. Mech. Astron.* 41 (2011) 816–823, doi:10.1360/132010-922.
- [29] J. Liu, G. Deng, Link prediction in a user-object network based on time-weighted resource allocation, *Phys. A* 388 (2009) 3643–3650.
- [30] W.P. Liu, L. Lü, Link prediction based on local random walk, *Eur. Phys. Lett* 89 (5) (2010) 58007.

- [31] L. Lü, C.-H. Jin, T. Zhou, Similarity index based on local paths for link prediction of complex networks, *Phys. Rev. E – Stat., Nonlinear Soft Matter Phys.* 80 (4) (2009) 046122.
- [32] L.Y. Lü, T. Zhou, Link prediction in complex networks: a survey, *Phys. A* 390 (2011) 1150–1170.
- [33] X. Ma, J.L. Liao, S.M. Djouadi, Q. Cao, LIPS: Link Prediction as a service for data aggregation applications, *Ad Hoc Netw.* 19 (2014) 43–58.
- [34] A. Papadimitriou, Panagiotis Symeonidis, Yannis Manolopoulos, Fast and accurate link prediction in social networking systems, *J. Syst. Softw.* 85 (9) (2012) 2119–2132.
- [35] A. Popescul, L. Ungar, Statistical relational learning for link prediction, in: *Proceedings of the International Workshop on Learning Statistical Models from Relational Data*, Vol. 149, Acapulco, Mexico, 2003, pp. 172–179.
- [36] M. Pujari, R. Kanawati, Supervised Rank Aggregation Approach for Link Prediction in Complex Networks, *WWW 2012 Companion*, Lyon, France, 2012, pp. 1189–1196.
- [37] E. Sherkat, M. Rahgozar, M. Asadpour, Structural link prediction based on ant colony approach in social networks, *Phys. A: Stat. Mech. Appl.* 419 (1) (2015) 80–94.
- [38] Y. Sun, R. Barbary, M. Gupta, C.C. Aggarwal, J. Han, Co-Author Relationship Prediction in Heterogeneous Bibliographic Networks, in: *Proceedings of 2011 International Conference on Advances in Social Networks Analysis and Mining, (ASONAM 2011)*, 2011, pp. 121–128.
- [39] A. Vidmer, A. Zeng, M. Medo, Y.C. Zhang, Prediction in complex systems: the case of the international trade network, *Phys. A: Stat. Mech. Appl.* 436 (2015) 188–199.
- [40] D.Q. Vu, A.U. Asuncion, D.R. Hunter, P. Smyth, Continuous-time regression models for longitudinal networks, *Advances in Neural Information Processing Systems* 24, in: *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, 2011, pp. 1–9.
- [41] X.J. Wang, X. Zhang, C.L. Zhao, Z. Xie, S.J. Zhang, D.Y. Yi, Predicting link directions using local directed path, *Phys. A: Stat. Mech. Appl.* 419 (2015) 260–267.
- [42] F. Xie, Z. Chen, J.X. Shang, X.P. Feng, J. Li, A link prediction approach for item recommendation with complex number, *Knowl.-Based Syst.* 81 (2015) 148–158.
- [43] L. Zhang, K. Hu, Y. Tang, Predicting disease-related genes by topological similarity in human protein-protein interaction network, *Cent. Eur. J. Phys.* 8 (4) (2010) 672–682.
- [44] Z.Z. Zeng, Ke-Jia Chen, Shaobo Zhang, Haijin Zhang, A link prediction approach using semi-supervised learning in dynamic networks, in: *Proceedings of Sixth International Conference on Advanced Computational Intelligence (ICACI)*, 2013, pp. 276–280.
- [45] T. Zhou, L. Lü, Y.C. Zhang, Predicting missing links via local information, *Eur. Phys. J. B* 71 (4) (2009) 623–630.